International Journal of Computer Vision 72(2), 159–178, 2007

DOI: 10.1007/s11263-006-8595-0

Implicit Meshes for Effective Silhouette Handling

SLOBODAN ILIĆ, MATHIEU SALZMANN AND PASCAL FUA *
Computer Vision Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
Slobodan.Ilic@epfl.ch
Mathieu.Salzmann@epfl.ch
Pascal.Fua@epfl.ch

Received September 27, 2005; Revised March 30, 2006; Accepted March 30, 2006

First online version published in July, 2006

Abstract. Using silhouettes in uncontrolled environments typically requires handling occlusions as well as changing or cluttered backgrounds, which limits the applicability of most silhouette based methods. For the purpose of 3-D shape modeling, we show that representing generic 3-D surfaces as implicit surfaces lets us effectively address these issues.

This desirable behavior is completely independent from the way the surface deformations are parametrized. To show this, we demonstrate our technique in three very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; reconstruction and tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations; reconstructing the shape of a human face parametrized in terms of a Principal Component Analysis model.

Keywords: 3-D modeling, silhouettes, tracking, deformable surfaces, implicit surfaces, bundle-adjustment

1. Introduction

Occluding contours are a key clue to recovering the shape of smooth and deformable objects. However, they are notoriously difficult to extract against potentially cluttered or changing backgrounds and in the presence of occlusions. As a result, it is standard practice to engineer the environment and to use multiple cameras, which limits the applicability of the resulting methods.

By contrast, we show that representing deformable 3-D shapes as implicit surfaces lets us successfully exploit occluding contours in monocular sequences filmed in difficult conditions, such as those depicted by Fig. 1. In these images, even the best edge-detection or background subtraction algorithms cannot be expected to work reliably. This is key to being able to exploit silhouettes in uncontrolled real-world situations. Furthermore, it also lets us effectively combine silhouette information with that provided by interest points, which can be tracked from image to image.

More specifically, we use *implicit meshes* (Ilić and Fua, 2003), which are implicit surfaces that closely approximate generic triangular 3-D meshes and deform in tandem with them. This formulation allows us to robustly compute the occluding contours on the 3-D surface as the solution of an ordinary differential equation (Rosten and Drummond, 2003). Their projections can then be used to search for the true image boundaries and deform the 3-D model so that it projects correctly.

This well-formalized approach yields a robust implementation that we demonstrate for monocular tracking of deformable 3-D objects in a completely automated

*This work was supported in part by the Swiss National Science Foundation

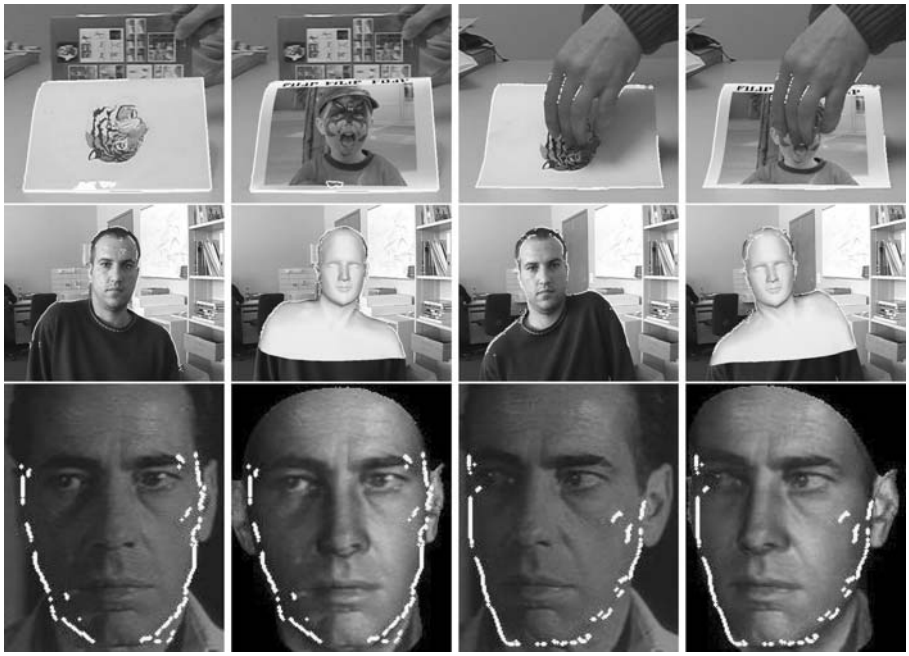


Figure 1. Detecting and using silhouettes for tracking and reconstruction from monocular sequences. The detected silhouette points are shown in yellow, or white if printed in black and white. First row: Tracking a deforming piece of paper with a tiger on it and replacing the tiger by a picture, which involves accurate 3-D shape estimation. This is done in spite of the moving book and the occluding hand. Middle row: Tracking the head and shoulders of a moving person. The reprojected 3-D model is shown as a shaded surface. Note that, even though the background is cluttered, we did not need to perform any kind of background subtraction. Bottom row: The face of a well known actor reconstructed from five very low quality 160×236 images obtained by digitizing an old celluloid film.

fashion: We start with a generic 3-D model of the target object. For the examples of Fig. 1, we use a simple planar triangulation for the deforming piece of paper, a custom-designed head-and-shoulder model for upper body tracking, and a morphable model (Blanz and Vetter, 1999) for facial reconstruction. We roughly position the model in the first image, find its occluding contours, and use them to search for the corresponding contours in the images. We then use the detected 2-D contours and the constraints they impose, along with frame-to-frame point correspondences, to refine the model's shape.

This approach is effective independently of the specific way the deformations are parametrized. As shown in Fig. 1, we validated the tracker in several very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; reconstructing and tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations (Moccozet and Magnenat-Thalmann, 1997); reconstructing the shape of a human face parametrized interms of a Principal Component Analysis model (Blanz and Vetter, 1999; Dimitrijević et al., 2004).

In the remainder of the paper, we first review related approaches and our earlier work (Ilić and Fua, 2003) on implicit meshes. We then show how we use them first to guide the search for silhouettes in the images, and second to enforce the corresponding differential constraints on the surface. Finally, we present our results in more details and quantify their accuracy.

2. Related Work

Occluding contours have long been known to be an excellent source of information for surface reconstruction, and sometimes the only available one when the surface slants away from the camera and makes it impractical to use other approaches such as feature point matching. This information has been put to very good effect by researchers, such as Sullivan et al. (1994), Vaillant and Faugeras (1992), Sullivan and Ponce (1998), Cross and Zisserman (2000), Szeliski

and Weiss (1998), Boyer and Berger (1997), Davis and Bobick (1998), Cipolla and Blake (1992), Terzopoulos and Metaxas (1991), among many others. In most of these works, the technique used to actually extract the occluding contours is relatively straightforward. It can be simple edge detection and linking (Canny, 1986), active contour models (Kass et al., 1988), or space carving (Kutulakos and Seitz, 2000). However, while perfectly appropriate in the context in which they are used, these methods would fail in the presence of cluttered and changing backgrounds.

Detecting occluding contours in such situations requires much more sophisticated algorithms. Recent color and texture-based segmentation algorithms (Paragios and Deriche, 1998; Rother et al., 2004; Carson et al., 2002) have proved very good at this. However, since they are essentially 2-D, it is not trivial to guarantee that the outlines they produce actually correspond to the target object's occluding contours.

A popular solution to this problem among researchers involved in tracking articulated or rigid objects is to model them using volumetric primitives whose occluding contours can be computed given a pose estimate. The quality of these contours can then be evaluated using either the chamfer distance to image edges (Gavrila and Davis, 1995) or more sophisticated measures (Sminchisescu and Triggs, 2003; Agarwal and Triggs, 2004). This quality measure can then be optimized to refine the pose, which has shown to be effective, but mostly in cases where either the subjects wear clothes that facilitate silhouette extraction or where the background is relatively simple. An alternative approach is to search for the true image boundaries in directions that are normal to them Kutulakos and Seitz (2000), Drummond and Cipolla (2002). This gives good results but has only been demonstrated for relatively simple shapes such as ellipsoids and truncated cones. The work we present here can be understood as a generalization of this approach to more complex surfaces that can deform in less predictable ways.

3. Implicit Meshes

In earlier work, we introduced *implicit meshes* (Ilić and Fua, 2003). They are implicit surfaces that are designed to closely approximate the shape of arbitrary triangulated meshes and to deform in tandem with them, as shown in Fig. 2. To convert a triangulated mesh into an implicit one, we attach a spherical or triangular implicit surface primitive, and corresponding field function f, to each facet. We then define the surface as the set

$$S(\mathbf{\Theta}) = \{\mathbf{x} \in R^3, F(\mathbf{x}, \mathbf{\Theta}) = T\} \quad , \tag{1}$$

where $F = \sum f_i$, $i = 1 \ldots N$ is the sum of the individual field functions, one for each of the N mesh facets, $\mathbf{\Theta}$ a set of parameters or *state vector* that controls the shape of the explicit mesh, and T a fixed isovalue.

A spherical primitive is created by circumscribing a sphere around the facet i so that the centers of the sphere and of the circle circumscribed around the facet coincide. In this case, f_i simply is

$$f_i(\mathbf{x}) = \exp\left(-k\left(r_i(\mathbf{x}) - r_i^0\right)\right) \qquad i = 1 \ldots N, \tag{2}$$

where $\mathbf{x}$ is a 3-D point, r_i is the Euclidean distance to the sphere's center, r_i^0 is the radius of the spherical primitive and k is a free coefficient defining the slope of the potential field function. For triangular primitives, we replace the Euclidean distance r_i by a piecewise

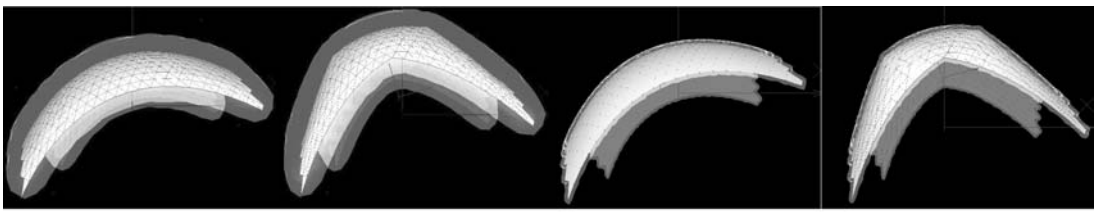


Figure 2. Approximating an explicit mesh by an implicit one. (a, b) Spherical implicit meshes wrapped around an explicit mesh and shown as being transparent. (c, d) Triangular implicit meshes. Note the much improved approximation.

polynomial C^1 function d_i that more accurately approximates the actual distance to the facet i. d_i is computed as the squared distance from the facet plane for points that project on the facet and as the squared distance from its edges and vertexes otherwise. f_i becomes

$$f_i(\mathbf{x}) = \exp\left(-k\left(d_i(\mathbf{x}) - d_0^2\right)\right), \tag{3}$$

which has almost the same form as before, but where d_0 now represents the thickness of the implicit surface and is the same for all facets.

Spherical primitives are best for relatively regular meshes because they are computationally inexpensive. Triangular primitives are more expensive but also more general and provide better surface approximations, especially when the explicit mesh is either irregular or low resolution. In any event, the method proposed in this paper is applicable to both since it only depends on the surface differentiability.

4. Silhouette Detection

As discussed earlier, given the estimated shape and pose of a 3-D model, our goal is to compute its 3-D occluding contours, project them into the image and use that projection as a starting guess to find the corresponding image boundaries, which should be the real silhouettes. In this section, we first show some of the problems involved in performing this task using traditional techniques. We then show that our implicit mesh formalism solves them and gives us cleaner and more consistent results, which can then be exploited to detect the right image boundaries.

4.1. Occluding Contours from Explicit Meshes

In the absence of the implicit surface formalism we propose, one of the most popular ways of finding occluding contours is to perform a visibility computation. For example, we can use OpenGL to project the model into the images, which lets us use the z-buffer to distinguish visible facets from hidden ones. The edges between these two sets of facets are then taken to be candidate occluding contours.

As shown in Fig. 3(b, c), the results of this procedure are heavily dependent on mesh resolution and the resulting contours are rarely as smooth as they should. Of course, more sophisticated heuristics would certainly yield improved results but we are not aware of any existing technique whose results are as clean and mesh-resolution independent as those of Fig. 3(d, e), which were obtained using our implicit surface formalism. As will be discussed in Section 6.1.2, this can have a dramatic influence on the quality of reconstruction results.

4.2. Occluding Contours and Ordinary Differential Equations

Occluding contours of implicit surfaces, such as the ones depicted by Fig. 3, can be found by solving

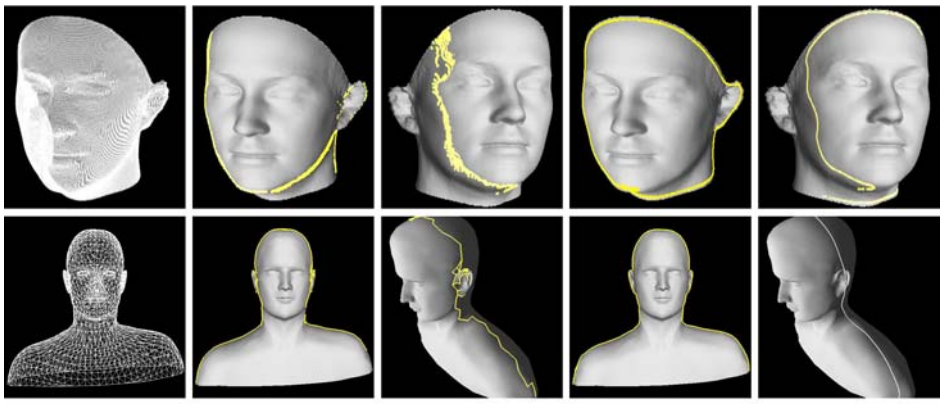


Figure 3. Occluding contours on explicit versus implicit meshes. (a) High resolution mesh of the face and low resolution mesh of the upper body. (b) Shaded model with edges at the boundary between visible and hidden facets overlaid in yellow. (c) The same edges seen from a different viewpoint (d, e) Shaded models with the occluding contour computed using implicit mesh, corresponding to views (b) and (c) respectively. Note the much greater smoothness and improved precision.

an Ordinary Differential Equation (ODE) as follows (Rosten and Drummond, 2003). Let $\mathbf{x}(s)$, $s \in [0, L]$ be a 3-D occluding contour on the implicit surface $S(\mathbf{\Theta})$ of Eq. 1, where L is its total length and s is the curvilinear abscissa.

1. $\mathbf{x}(s)$ is on the surface and therefore

$$F(\mathbf{x}(s), \mathbf{\Theta}) = T \ ; \tag{4}$$

2. the line of sight is tangential to the surface at $\mathbf{x}(s)$, which implies

$$(\mathbf{x}(s) - \mathbf{COpt}) \bullet \bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta}) = 0 \ . \tag{5}$$

Differentiating Eqs. (4) and (5) with respect to the s abscissa yields

$$0 = \frac{\partial F(\mathbf{x}(s), \mathbf{\Theta})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(s)}{\partial s} = \bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta}) \frac{\partial \mathbf{x}(s)}{\partial s} \tag{6}$$

$$0 = \frac{\partial \mathbf{x}(s)}{\partial s} \bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta}) + (\mathbf{x}(s) - \mathbf{COpt}) H(\mathbf{x}(s), \mathbf{\Theta}) \frac{\partial \mathbf{x}(s)}{\partial s} \tag{7}$$

where $H(\mathbf{x}(s), \mathbf{\Theta})$ is the Hessian matrix of F, $\bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta})$ its gradient vector and **COpt** the optical center of the camera depicted by Fig. 4. Substituting Eq. (6) into Eq. (7) allows us to eliminate the first term, which yields

$$(\mathbf{x}(s) - \mathbf{COpt}) H(\mathbf{x}(s), \mathbf{\Theta}) \frac{\partial \mathbf{x}(s)}{\partial s} = 0. \tag{8}$$

Note that Eqs. (6) and (8) imply that the vector $\frac{\partial \mathbf{x}(s)}{\partial s}$ is perpendicular to both $\bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta})$ and $(\mathbf{x}(s) - \mathbf{COpt}) H(\mathbf{x}(s), \mathbf{\Theta})$. We can therefore write

$$\frac{\partial \mathbf{x}(s)}{\partial s} \propto ((\mathbf{x}(s) - \mathbf{COpt}) H(\mathbf{x}(s), \mathbf{\Theta})) \times \bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta}). \tag{9}$$

Since by definition of the curvilinear abscissa, $\| \frac{\partial \mathbf{x}(s)}{\partial s} \| = 1$, this implies that $\mathbf{x}(s)$ is a solution of the ODE

$$\frac{\partial \mathbf{x}(s)}{\partial s} = \frac{((\mathbf{x}(s) - \mathbf{COpt}) H(\mathbf{x}(s), \mathbf{\Theta})) \times \bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta})}{\| ((\mathbf{x}(s) - \mathbf{COpt}) H(\mathbf{x}(s), \mathbf{\Theta})) \times \bigtriangledown F(\mathbf{x}(s), \mathbf{\Theta}) \|}. \tag{10}$$

We solve this ODE in vector space using a 4th order Runge-Kutta scheme, which involves computing the Hessian matrix. In theory, this requires the distance

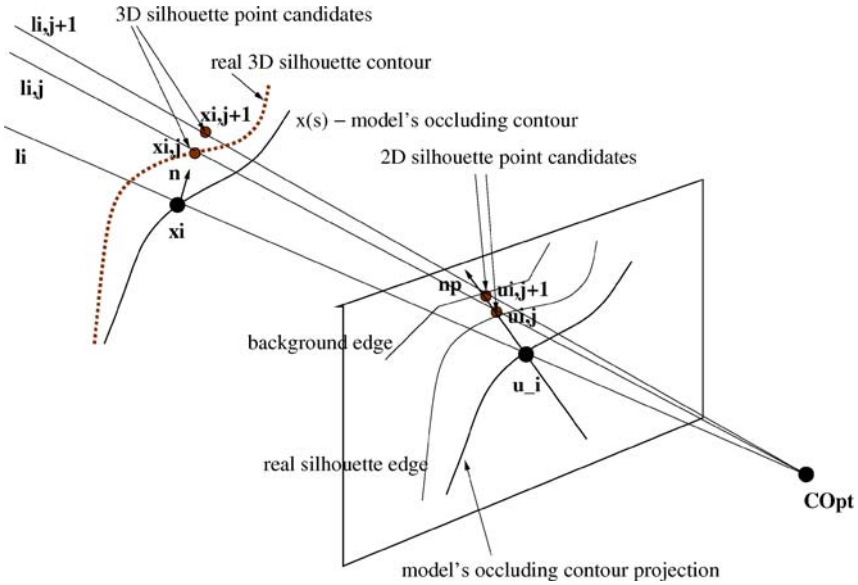


Figure 4. Finding multiple silhouette edge points in the image. Notations are defined in Section 4.3.

functions to be C^2, which is strictly true when using spherical primitives but not triangular ones. In the latter case, the distance function is C^1 everywhere and C^2 except at facet edges and vertexes. In practice, because the surfaces are smooth, this has not caused us any problems. This approach results in the very clean contours of Fig. 3(d, e) that are quite insensitive to the resolution of the mesh used to compute them.

Obtaining those results requires an initial condition for the ODE, which is given as $\mathbf{x}(0)$, a 3-D starting point on the occluding contour. To find it, we use a visibility algorithm similar to the one described in Section 4.1. We select one *single* vertex of the explicit mesh that is very likely to be an occluding vertex, and its location is heuristically determined to be close to the wanted beginning of the occluding contour. We then project it onto the implicit mesh and search in the neighborhood of the projection for a point that satisfies the two constraints given by Eqs. (4) and (5). Note that this is very different from the approach of Section 4.1 because, since we only need one 3-D point, we can impose very tight constraints and thus ensure that it really is on the occluding contour. The length of the resulting contour is a function of the number of Runge-Kutta steps and of the corresponding step size. In the tracking examples presented in the results section, the expected length of the occluding contour does not vary dramatically. We therefore fix the step size and choose the number of steps so that the resulting contour length is appropriate in the first frame.

4.3. Finding Silhouette Edges in the Image

Given a 3-D occluding contour $\mathbf{x}(s)$ computed as described above, we project it into the image and look for the true silhouette edge in a direction normal to its 2-D projection as depicted in Fig. 4. This is geometrically consistent because, at a silhouette point $\mathbf{x}_i \in \mathbf{x}(s)$, $s \in [0, L]$, the 3-D surface normal $\mathbf{n}$ is perpendicular to the line of sight $\mathbf{l}_i$ and, as a result, projects to the normal $\mathbf{n}_p$ of the 2-D contour.

In other words, at each point $\mathbf{u}_i$ of the 2-D projection, we simply have to perform a 1-D search along a scan-line for the true edge location and we are back to the old edge detection problem, but in a much simpler context than usual. We use a technique that has proved effective for edge-based tracking (Vacchetti et al., 2004; Drummond and Cipolla, 2002): Instead of selecting one arbitrary gradient maximum along the scan-line, we select multiple gradient maxima resulting in several potential silhouette edge points $\mathbf{u}_i^j$ and corresponding lines of sight $\mathbf{l}_i^j$ for each $\mathbf{x}_i$. Along these new lines of sight, we could choose the $\mathbf{x}_i^j$ where the line is closest to the surface as the most likely point where the surface should be tangent to the line of sight. However, this involves a computationally expensive search along the line of sight. In practice, as shown in Fig. 4, (a) simpler and equally effective approach is to take each $\mathbf{x}_i^j$ to be the point on $\mathbf{l}_i^j$ that is at the same distance from the optical center as the original $\mathbf{x}_i$. These $\mathbf{x}_i^j$ are then used to enforce silhouette constraints, as explained in Section 5.

5. Fitting Implicit Mesh 3-D Models

Silhouettes are a key clue to surface shape and deformation in monocular sequences, but they are also a sparse one since they are only available at a few image locations. For objects that are somewhat textured, point correspondences between interest points in pairs of images complement them ideally. They can be established best where silhouettes are least useful, that is on the parts of the surfaces that are more or less parallel to the image plane.

In the following section, we show that our formalism lets us effectively combine these two information sources. Given a set of correspondences and silhouette points, we fit our model to the data by minimizing a set of *observation equations* in the least-squares sense. To this end we use the Levenberg-Marquardt algorithm (Press et al., 1992), which we have found very effective at solving these kinds of non-linear optimization problems. At each iteration, we recompute the occluding contours and corresponding silhouette points in the image using the technique of Section 4.2. To increase robustness, we introduce an iterative reweighted least-squares scheme that allows us to discount erroneous correspondences or silhouette points.

As we will see, the silhouette-based constraints are best expressed in terms of the implicit surface formalism while it is simpler to formulate the correspondence-based ones using traditional triangulated meshes. Recall from Section 3 that both the implicit surface and the underlying explicit one deform in tandem when the state vector changes. As a result, we can simultaneously use the implicit formalism when dealing with silhouettes and the explicit one when dealing with correspondences as needed to simplify our

implementation. We view this as one of the major strengths of our approach.

5.1. Least Squares Framework

We use the image data to write n_{obs} observation equations of the form

$$Obs(\mathbf{x}_i, \mathbf{\Theta}) = \epsilon_i \ , \ 1 \leq i \leq n_{\text{obs}}, \tag{11}$$

where $\mathbf{x}_i$ is a data point, $\mathbf{\Theta}$ the state vector of Eq. (1), Obs a differentiable function whose value is zero for the correct value of $\mathbf{\Theta}$ and completely noise free data, and ϵ_i is treated as an independently distributed Gaussian error term. We then minimize $v^T P v$, where $v = [\epsilon_1, \ldots, \epsilon_{n_{\text{obs}}}]$ is the vector of residuals and P is a diagonal weight matrix associated with the observations. If estimates of the reliability of each source of information were available, a minimum variance solution could be obtained by weighting each source of information according to these estimates. Unfortunately, such confidence measures are rarely available and we set these weights as described below.

Because there are both noise and potential gaps in the image data, we add a regularization term E_D that forces the deformations to remain smooth. Its exact formulation depends on the chosen parametrization and will be discussed in more details in the result section. We therefore write the total energy that we minimize as

$$E_T = \sum_{i=1}^{nobs} c_{type_i} \left(Obs^{type_i}(\mathbf{x}_i, \mathbf{\Theta}) \right)^2 + E_D \ , \tag{12}$$

where $type_i$ is the type of observation i, Obs^{type_i} is the corresponding observation function, and the c_{type_i} are weights that are taken to be equal for all observations of the same type. We introduce this heuristic because the iteration steps of the Levenberg-Marquardt algorithm (Press et al., 1992) we use to optimize E_T are computed using the Jacobian matrix formed by concatenating the ∇Obs^{type_i} gradient vectors of the Obs^{type_i} functions. Given that they are computed using different data types, the magnitude of these gradients are not be commensurate in general. To ensure that the minimization proceeds smoothly, they must be scaled so that, initially at least, the magnitudes of the gradient terms corresponding to the different observation types have appropriate relative values. This is achieved by writing

$$c_{type} = \frac{\lambda_{type}}{G_{type}}, \tag{13}$$

where

$$G_{type} = \frac{\sqrt{\sum_{1 \leq i \leq nobs, type = type_i} \| \nabla Obs^{type_i}(\mathbf{x}_i, \mathbf{S}) \|^2}}{n_{type}} \ , \tag{14}$$

and λ_{type} is a user supplied coefficient between 0 and 1 that indicates the relative importance of the observation types. In practice we give the same influence to the two observation types by choosing $\lambda_{silh} = \lambda_{corr} = 0.5$.

We now turn to the description of Obs^{silh} and Obs^{corr}, the functions that handle silhouettes and correspondences, the two data types we actually use here.

5.2. Silhouettes

In Section 4, we showed how to use our formalism to associate 2-D image locations to 3-D surface points that lie on the occluding contours. Recall from Section 4.3 that, for each 3-D point $\mathbf{x}_i$ on the model's occluding contour, we defined a corresponding set of candidate occluding points on the target object, the $\mathbf{x}_i^j$ depicted by Fig. 4. If the shape and pose of the 3-D model were perfect, one of them should be on the surface and therefore satisfy $F(\mathbf{x}_i^j, \mathbf{\Theta}) = T$, where F is the field function of Eq. 1, and T the isovalue defined in the same equation. During the optimization, this will in general not be true. We enforce this constraint by introducing for each $\mathbf{x}_i^j$ a *silhouette function* of the form

$$Obs^{silh}\left(\mathbf{x}_i^j, \mathbf{\Theta}\right) = w_i^j \left(F\left(\mathbf{x}_i^j, \mathbf{\Theta}\right) - T \right), \tag{15}$$

where w_i^j is a weight associated to the observation. It is taken to be inversely proportional to its distance to the line of sight $\mathbf{l}_i^j$ depicted by Fig. 4. As the total energy E_T of Eq. (12) is minimized, the $Obs^{silh}(\mathbf{x}_i^j, \mathbf{\Theta})$ will collectively decrease in the least-squares sense. Eventually, for each $\mathbf{x}_i$, only one of these candidates, $\mathbf{x}_i^{best}$, will end up being close to $\mathbf{l}_i$ and having a large w_i^j weight, while the others will be ignored. x_i^{best} will also become closer and closer to actually being on the surface and, because it minimizes the distance to the surface along $\mathbf{l}_i$, the normal at the closest surface point will be perpendicular to the line of sight. Thus, $\mathbf{x}_i^{best}$ will eventually tend to satisfy the two conditions introduced

in Section 4.2 that characterize an occluding contour point.

5.3. Correspondences

Given a calibrated n image-sequence, such as the one depicted by Fig. 5, we use 2-D point correspondences in two or more consecutive images. We begin with the case of two images and then expand our approach to the full sequence.

Let us first consider the pair of consecutive images i and $i+1$, with $1 \leq i < n$. Let p_i^k be a 2-D point in image i and p_{i+1}^k a corresponding one in image $i+1$, found using a simple correlation-based algorithm. We back-project p_i^k to the 3-D surface and re-project the resulting 3-D point into image $i+1$, yielding $\hat{p}_{i+1}^k = \psi(p_i^k, \mathbf{\Theta})$, where ψ is known as the transfer function depending on the state vector $\mathbf{\Theta}$. We can now take the correspondence function to be

$$\begin{aligned} Obs^{corr}\left(\{p_i^k, p_{i+1}^k\}, \mathbf{\Theta}\right) &= w_i^k \left\| \hat{p}_{i+1}^k - p_{i+1}^k \right\| \\ &= w_i^k \left\| \psi\left(p_i^k, \mathbf{\Theta}\right) - p_{i+1}^k \right\|, \end{aligned} \tag{16}$$

the Euclidean distance in image $i+1$ between the corresponding point and the reprojected one multiplied by a weight w_i^k. Since some of the correspondences may be erroneous, as in the silhouette case, this weight is taken to be inversely proportional to the corresponding residual at the end of the previous minimization. As a result, after several minimizations, false correspondences tend to be ignored.

Note that the simplest and fastest way of back-projecting p_i^k to the 3-D surface is to use OpenGl and the graphics hardware of our machines to find the facet that is traversed by the line of sight it defines. Therefore, in our implementation, when computing $Obs^{corr}(\{p_i^k, p_{i+1}^k\}, \mathbf{\Theta})$ and its derivatives, we use the explicit representation of the model instead of the implicit one.

In practice and depending on the application, the p_i^k points are either obtained by regularly sampling the 3-D model projection into image i or are taken to be interest points detected using the Harris corner detector. Note that we can use correspondences $\{p_i^k, p_{i-1}^k\}$ in image $i-1$ in exactly the same way since the ψ transfer function can be defined between any two images. In

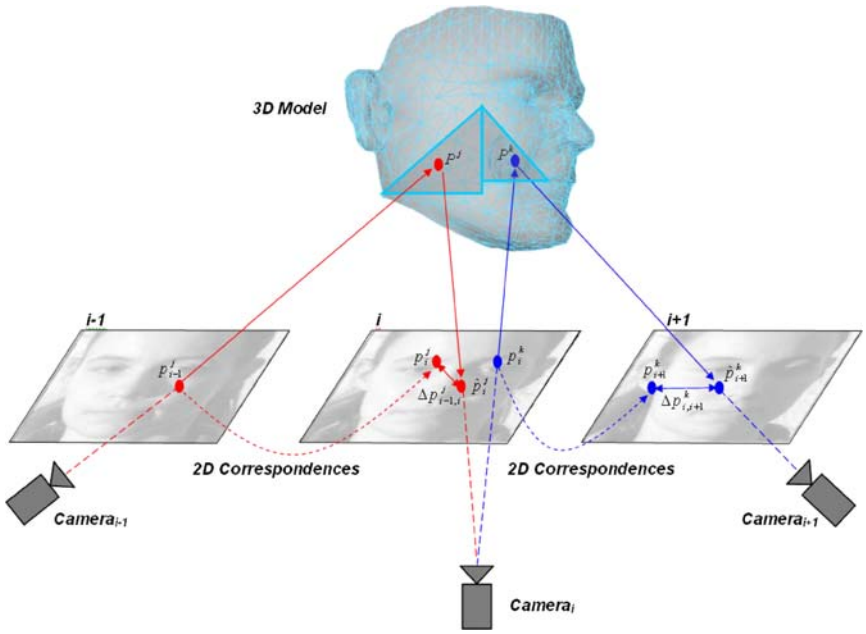


Figure 5. Exploiting correspondences. First, the model, a face in this case, is projected into the reference image, labeled as image i. Depending on the application, either the projection is densely sampled or the sample points are taken to be interest points detected using the Harris corner detector. Then a correlation based algorithm is used to establish correspondences in the following $i+1$ or in the preceding $i-1$ and following $i+1$ images. The points from the reference image are back-projected to the 3-D model and reprojected into the other images. The sum of the squares of the distances between these back-projections and the corresponding points is minimized in terms of only model shape parameters or in terms of model and the camera parameters in case of uncalibrated sequences.

this way, as shown in Fig. 5 we can use one set of p_i^k sample points to create observations that exploit the correspondences in the image triplet $i-1, i$, and $i+1$.

When dealing with a full image sequence, we repeat this operation for all consecutive image triplets. In other words, we sample the middle image, compute correspondences in the other two, and use all these correspondences to create additional correspondence observations. In this way, we can simultaneously exploit the shape information present in the whole sequence. Since correspondences can be expected to be noisy and sometimes erroneous, this increases our algorithm's robustness. All frames play equivalent roles and there is no need to arbitrarily choose a reference one.

5.4. *Uncalibrated Video Sequences*

When handling video sequences acquired using an unknown camera in which the object moves, we fix the camera parameters to reasonable values and estimate the frame-to-frame rotations and translations in addition to the shape. More specifically, we introduce for each frame i a vector C_i that contains the 3 rotations and translations with respect to an absolute coordinate system. Let $\mathbf{C} = \{C_1, \ldots, C_n\}$ be the vector of all such camera parameters. The observation equation of Eq. (11) becomes

$$Obs(\mathbf{x}_i, \mathbf{\Theta}, \mathbf{C}) = \epsilon_i \ , \ 1 \leq i \leq n_{\text{obs}}. \qquad (17)$$

We therefore end up minimizing the *total energy*

$$E_T = \sum_{i=1}^{nobs} c_{type_i} \left(Obs^{type_i}(\mathbf{x}_i, \mathbf{\Theta}, \mathbf{C}) \right)^2 + E_D, \quad (18)$$

with respect to both $\mathbf{\Theta}$ and $\mathbf{C}$. In practice, an approximate value of $\mathbf{C}$ has to be supplied for initialization purposes. To this end, we choose a frame, say i_0 with $0 \leq i_0 \leq n$, as a reference frame and we compute $\mathbf{C_{i_0}}$ so that the model's initial projection roughly aligns with the target object. In the case of the faces shown in the result section, this is done by manually selecting five feature points in the reference frame and using POSIT (David et al., 2002). The $\mathbf{C_i}$ vectors for frames before and after i_0 are initially taken to be equal to $\mathbf{C_{i_0}}$ and then refined during the optimization.

In Dimitrijević et al. (2004), we showed that minimizing the objective function of Eq. (18) is a well conditioned least-squares problem, that yields reliable estimates of both camera position and shape, if we use enough correspondences. In practice, we therefore first estimate shape and camera parameters using correspondences only. For that purpose we developed an optimization schema where the shape parameters are progressively added as explained in Dimitrijević et al. (2004). We then refine the shape using the silhouettes, which require reasonably good estimate of the camera positions to be meaningful. Note that in our approach, there never is an explicit association between 2-D sample points in the images and specific vertexes or facets of the 3-D models. Instead, these associations are computed dynamically during the minimization and can change. As a result, we do not need to fix the $\mathbf{C_{i_0}}$ parameters during the optimization. Instead, they are refined in the same way as all the others and a rough initial estimate suffices. Again, all frames end up playing the same role and the reference one is only used for initialization purposes.

6. Results

In this section, we demonstrate the applicability of our method to the shape recovery of rigid and deformable objects from monocular video sequences. We also show that our formalism applies independently of the specific parametrization used to represent the shape and its deformations. To this end, we use as examples morphable face models (Blanz and Vetter, 1999), upper body models whose deformations are expressed in terms of Dirichlet Free Form Deformations (Moccozet and Magnenat-Thalmann, 1997), and ordinary triangulated meshes parametrized in terms of the 3-D coordinates of their vertexes. In all three cases, we validate our results using either laser scanner data or side views that were *not* used for reconstruction purposes.

6.1. *Rigid Objects*

Recovering the shape not only of human faces but also of the complete human upper body—head, neck, and shoulders—is a good proving ground for our technique because all the tools required to produce complete models are available to us. We use uncalibrated video sequences in which the subject, or the camera, move rigidly from frame to frame. We supply approximate values for the intrinsic parameters and use the

technique outlined in Section 5.4 to recover the motion and a first shape estimate using correspondences only. We then use the silhouette information to refine it.

6.1.1. Face Reconstruction. We use morphable face models (Blanz and Vetter, 1999), which are stored as triangulated meshes with 75292 vertexes. Given such a mesh, let us consider S, the *shape vector* obtained by concatenating the X, Y and Z coordinates of all its vertexes. A database of 3-D faces was used to compute the shape vectors for 200 people and Principal Component Analysis to approximate them as

$$S = \bar{S} + \sum_{i=1}^{99} \alpha_i S_i \quad , \tag{19}$$

where $\bar{S}$ represents an average face model, the S_i are orthogonal vectors, and the α_i are weights that can be varied to create a whole range of new faces. We therefore take Θ, the state vector of Eq. (1), to be the set of all these weights, initially set to zeros.

We take the E_D regularization term of Eq. (12) to be

$$E_D = \sum_{i=1}^{99} \frac{\alpha_i^2}{\sigma_{S_i}^2} \tag{20}$$

where the σ_{S_i} are the eigenvalues of the shape covariance matrix provided with the model (Blanz and Vetter, 1999). To effectively minimize the total energy E_T, we developed an optimization schedule in which the number of state variables, that are allowed to vary, progressively increases (Dimitrijević et al., 2004).

Using correspondences alone yields the results depicted by Fig. 6(b) for a seven-image sequence in which the subject faces the camera. The reconstruction is quite accurate in the front but the cheeks are too narrow. Since correspondences can be expected to be less and less accurate on the side of the face where the surface slopes away from the camera, this is not particularly surprising. However, using both correspondences and silhouettes yields the improved reconstruction of Fig. 6(c). It is now accurate both in front and on the side, as evidenced by the fact that its occluding contours accurately match the true silhouettes.

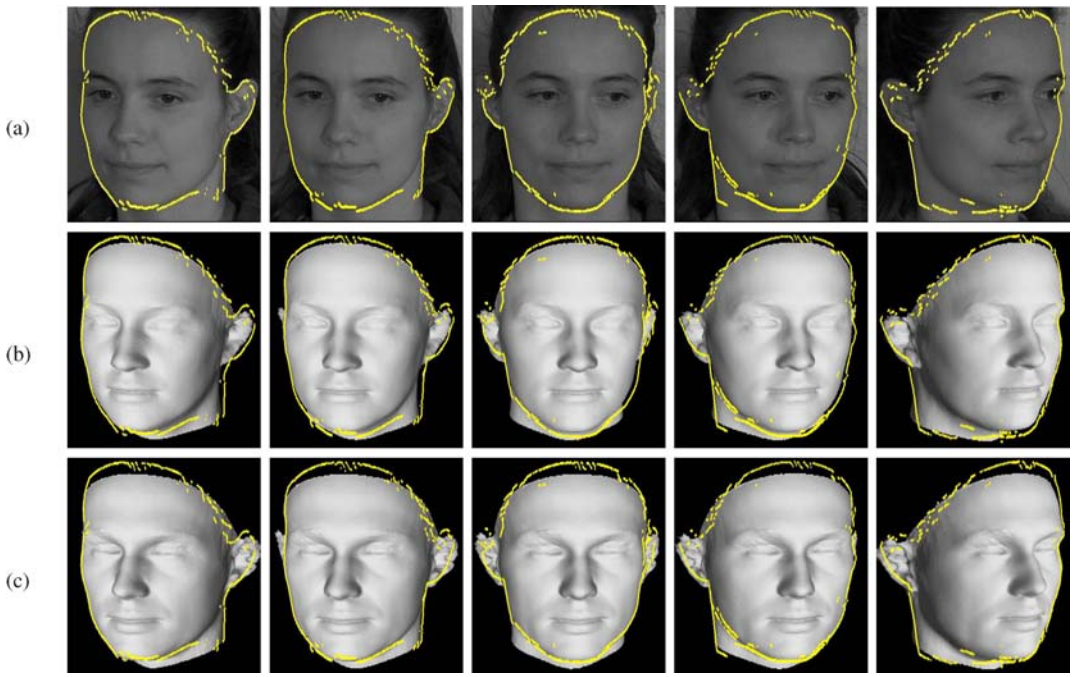


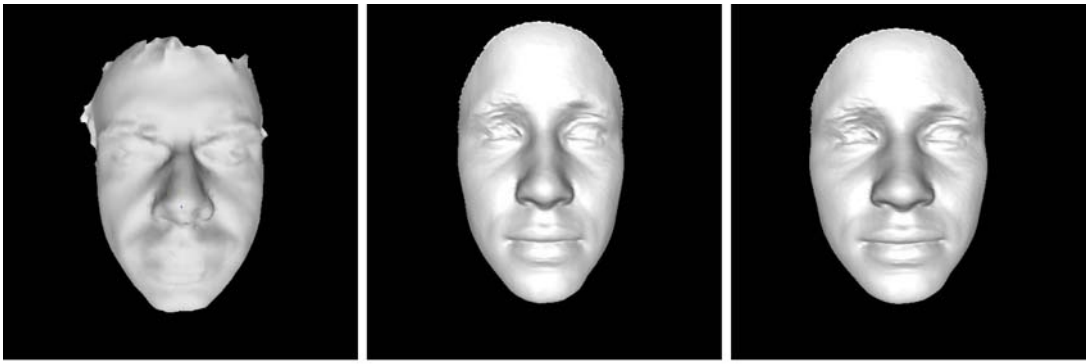
Figure 6. Head modeling using morphable face models. (a) Five of the seven images we used. (b) Reconstructed model using correspondences only. (c) Reconstructed model using both correspondences and silhouettes. The white outlines are the silhouette edges detected by applying the technique of Section 4.3 to the model of (b). We overlay them on the original images of row (a) to show that they correspond to the true silhouettes and to those of row (b) to highlight the inaccuracy of the reconstruction in the vicinity of those silhouettes, as well as to those of row (c) to indicate improved accuracy.

Figure 7. Head modeling using morphable face models. (a) Four images from a short video sequence. (b) Corresponding Canny edge images. (c) Reconstructed model using both correspondences and silhouettes. (d) Textured face model. As in Fig. 6, we overlay the extracted silhouette edges on the images to show that the model's occluding contours match the true silhouettes.

Figure 7 depicts a second example. The lighting conditions were relatively poor, making the edges difficult to extract, as evidenced by the output of the Canny edge detector with standard settings. Nevertheless, the model's occluding contours still superpose correctly with the true silhouettes.

To quantify the quality of these results, we have used a Minolta^{tm} laser scanner to acquire models of the subjects of Figs. 6 and 7. Figure 8(a) depicts one of them. When performing these experiments, our goal was to demonstrate that we could achieve automated shape recovery from completely uncalibrated video sequences. In theory, given high precision matches, we should be able to recover both intrinsic parameters and camera motion (Hartley and Zisserman, 2000). In practice, however, we must be prepared to deal with the potentially poor quality of the point matches. Therefore, we use an approximate value for the focal length and assume that the principal point remains in the center of the image, which results in a reconstruction that can only be accurate up to an affine transform (Fua, 2000). To compare the video-based reconstructions to the scans, we therefore compute an affine transform that brings the reconstructed model as close as possible to the scan. This is done by least-squares minimization and results in deformations of the model such as the one depicted by Fig. 8(c). In practice, if the intrinsic parameters we supply are reasonable, these deformations are close to being simple scalings along the x, y, and z axes. In Fig. 9, we plot for both subjects the proportion of the 3-D points in the laser scan that are within a given distance of the reconstructed model, after it has been deformed using the affine transform discussed above. The median distances are 0.98 mm and 0.73 mm respectively.

Our scanner has a theoretical precision of around 0.3 mm. However, this does not take into account some of the artifacts it produces, such as those that occur when stitching together several scans of the target object seen from different viewpoints. A median distance between two independent reconstructions under 1.0 mm is therefore a good indication that they are



(a) (b) (c)

Figure 8. Comparing against Laser Scanner Data. (a) Laser scan of the subject of Fig. 7. (b) Shaded view of the corresponding model reconstructed using the video sequence. (c) Deformed model after an affine transformation has been applied to bring it as close as possible to the laser scan of (a). Note that the deformation with respect to (b) is mild and close to being a simple scaling along the x, y, and z axes.

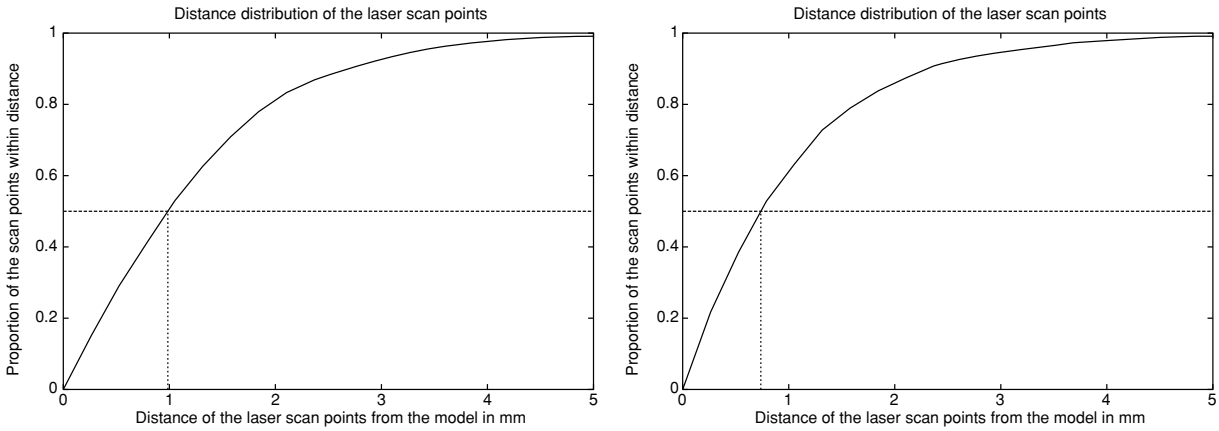


Figure 9. Quantitative evaluation of the facial reconstructions. (a) For the subject of Fig. 6, proportion of the 3-D points in the laser scan that are within a given distance of the reconstructed model, after it has been deformed using an affine transform. The median distance of 0.98 mm is denoted by the dotted lines. (b) Same thing for the subject of Fig. 7 with a median distance of 0.73 mm.

consistent with each other and presumably both close to the ground truth.

6.1.2. Upper Body Reconstruction. The morphable models we used above are outstanding to model faces but were never designed to model the whole upper body. To this end, and to demonstrate our technique in a different context, we use the model depicted by the bottom row of Fig. 3. Even though its resolution is relatively low, parameterizing it in terms of the 3-D coordinates of its vertexes would result in too many degrees of freedom for effective optimization. Instead, we use the Dirichlet Free Form Deformations (DFFD) formalism (Moccozet and Magnenat-Thalmann, 1997; Ilić and Fua, 2002). DFFDs let us control the model's shape using a small number of *control points*. This, in turn, allows us to formulate the shape recovery problem in terms of minimizing the energy of Eq. (12) with respect to a comparatively small number of parameters. We briefly describe the DFFD formalism below and refer the interested reader to our earlier publication (Ilić and Fua, 2002) for additional details.

Let $\{P_1, \ldots, P_N\}$ be the N vertexes of the upper body triangulation of Fig. 3 and let $\{Q_1, \ldots, Q_M\}$ be the set of control points of Fig. 10(a), with $M \ll N$. Here, the control points are taken to be a subset of the mesh vertexes. Translating each Q_j control point by

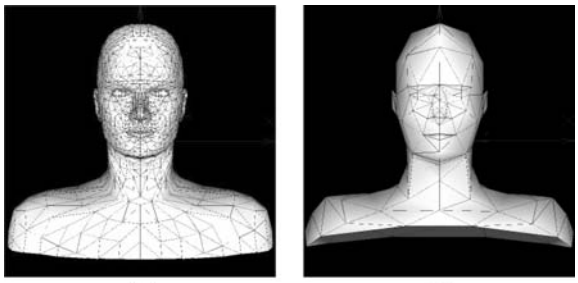


Figure 10. DFFD formalism. (a) The control points are taken to be a subset of the vertexes of the depicted model. (b) These control points are triangulated to form a control mesh.

ΔQ_j results in a displacement of vertex P_i by

$$\Delta P_i = \sum_{j=1}^{M} s_j^i \Delta Q_j \quad , \tag{21}$$

where the s_j^i are *Sibson coordinates* (Moccozet and Magnenat-Thalmann, 1997) that express the respective influence of the control point Q_j on the vertex P_i. In practice, these coordinates are computed once when the control points are chosen and do not need to be recomputed thereafter. Furthermore, for any given P_i, they are non-zero only for a small subset of neighboring control points. Therefore, we can take the state vector of Eq. (1) to be

$$\mathbf{\Theta} = \{\Delta Q_1, \ldots, \Delta Q_M\} \quad , \tag{22}$$

the concatenation of the individual control point displacements, initially set to zeros. Given a specific instantiation of $\mathbf{\Theta}$, the coordinates of the mesh vertexes become

$$\begin{bmatrix} P_1' \\ P_2' \\ \vdots \\ P_N' \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{bmatrix} + \begin{bmatrix} \sum_{j=1}^{M} s_j^1 \Delta Q_j \\ \sum_{j=1}^{M} s_j^2 \Delta Q_j \\ \vdots \\ \sum_{j=1}^{M} s_j^N \Delta Q_j \end{bmatrix} \quad . \tag{23}$$

Since we expect the deformation between the initial shape and the original one to be smooth, we triangulate the control points of Fig. 10(a) to create the *control mesh* of Fig. 10(b). We then take the E_D regularization term of Eq. (12) to be the sum of the square of the derivatives of the ΔQ_j displacements across this control mesh. By treating its facets as C^0 finite elements, we can approximate E_D as the quadratic form

$$E_D = \mathbf{\Theta}^t K \mathbf{\Theta} \quad ,$$

where K is a very sparse stiffness matrix.

In the example of Fig. 11 we use as input an initially uncalibrated 14–frame video sequence in which the camera is fixed and the subject holds still while sitting on a rotating chair. We first performed model-driven bundle-adjustment (Fua, 1999) to compute the relative motion and, thus, register the images. We then optimized the total energy E_T of Eq. (12) with respect to $\mathbf{\Theta}$, which yields the results depicted by the bottom two rows of the figure. Once again the occluding contours of the recovered model match the true silhouettes quite accurately.

As for the faces of Section 6.1.1, we laser scanned the subject's upper body and plot the distribution of distances between the scan and the reconstruction. Fig. 12(a) depicts the distribution of these distances for the face only and Fig. 12(b) for the whole model. The median distances, 1.2 mm and 2.03 mm respectively, are higher than those we obtained when using the morphable models. This makes sense both because the model we use here is coarser and because there is very little texture on the subject's sweater, thus making it difficult to establish correspondences there.

To highlight some of the problems encountered when using standard explicit meshes as opposed to implicit ones we advocate here, we replaced the 3-D occluding contour points detected by solving the ODE of Section 4.2 by those found using the technique of Section 4.1. As shown in Fig. 3(b, c) they tend to be inaccurate and discontinuous in 3-D. Furthermore, when not using the implicit surface formalism, we have to take the distance of an occluding contour point to the surface to be the orthonormal distance to the closest facet. This introduces non-differentiabilities in the objective function when the closest facet changes. As shown in 11(b), these two problems result in a much degraded reconstruction.

6.2. *Tracking Deformable Objects*

We now turn to tracking the motion of deformable 3-D objects in monocular sequences where the camera is fixed. We manually position the model in the first frame of each sequence. We then minimize the total energy E_T defined for each two consecutive frames with respect to the state vector $\mathbf{\Theta}$ using both

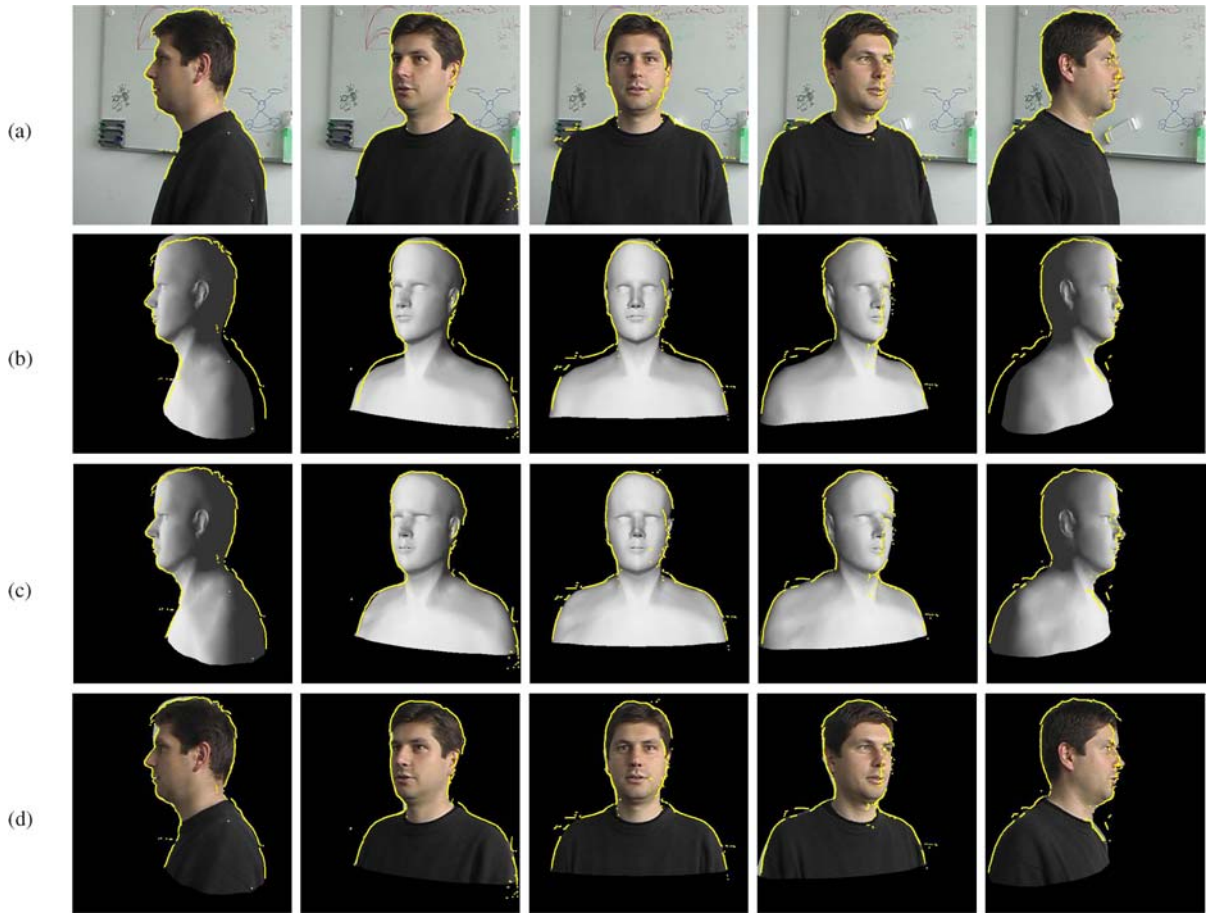


Figure 11. Reconstruction from an uncalibrated video sequence. (a) 5 of 14 images from a short video sequence. As before, we overlay in yellow the extracted silhouette edges on the images. (b) Using a standard explicit mesh to fit the low resolution upper body model of Fig. 3 to correspondence and silhouette data. Because the 3-D occluding contours are not accurately recovered using this formalism, the resulting shape is inaccurate and its projections do not adequately match the detected silhouettes. (c) Using our implicit meshes dramatically improves the quality of the result. (d) Texture-mapped version of the recovered 3-D model.

correspondences between two frames and occluding contours. This procedure is recursively repeated for all consecutive subsequent frames.

6.2.1. Head and Shoulders Tracking. We begin with the model of Fig. 13, which was built in exactly the same way as the one of Fig. 11, that is by asking the subject to remain still while we rotated the chair he sat on. We then acquired the sequences of Figs. 14 and 15 where he now faces the camera, wears different clothing, and, most importantly, moves his head and shoulders.

As in Section 6.1.2, the model is parametrized in terms of the DFFD state vector of Eq. (22). In the first row of Figs. 14 and 15, we show several frames from the original sequences. Since the subject keeps on facing the camera, the sequences provide much less reliable information on the shape of the head than the one of Fig. 13 where the head rotates in front of the camera. Since our DFFD parametrization does not include a suitable regularization term for plausible facial motions, using this unreliable information degrades the quality of the head reconstruction rather than improves it. We therefore constrain the head motion to be rigid and allow the control points that influence the rest of the model to move freely, which results in the models depicted by the second rows of both figures. In the bottom rows, we texture-map and re-project the deformed 3-D models into the original images. Note that we used the texture of the light blue T-shirt worn by the subject in the video of Fig. 13 used to build the static model, as opposed to the dark blue sweater

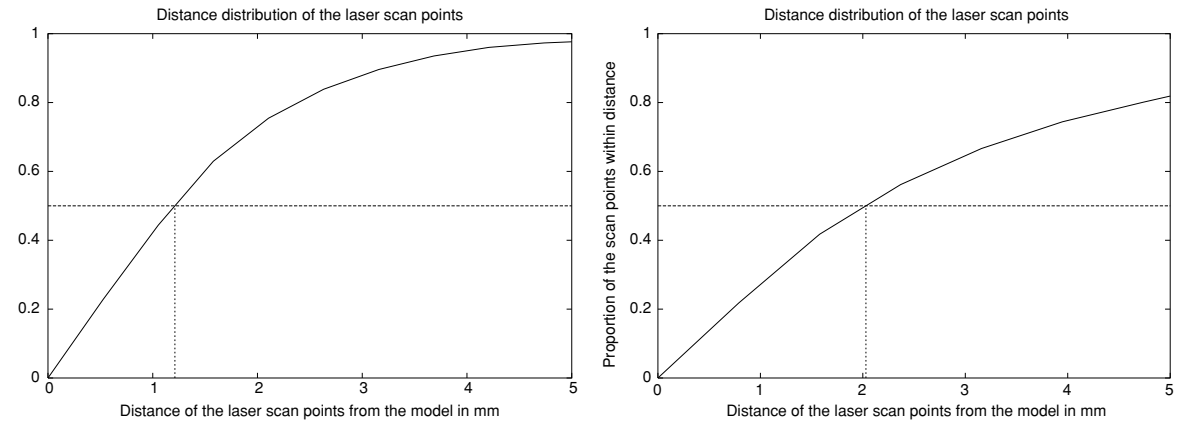


Figure 12. Quantitative evaluation of the upper body reconstruction of Fig. 11. (a) Proportion of the 3-D points in the laser scan that correspond to the face area and are within a given distance of the reconstructed model, after it has been deformed using an affine transform. The median distance of 1.2 mm is slightly larger than those of Fig. 9 because the model we use here is coarser. (b) Proportion for all the points in the laser scan. The median distance of 2.03 mm is even larger because there are few correspondences on the subject's torso, making the reconstruction less accurate there.

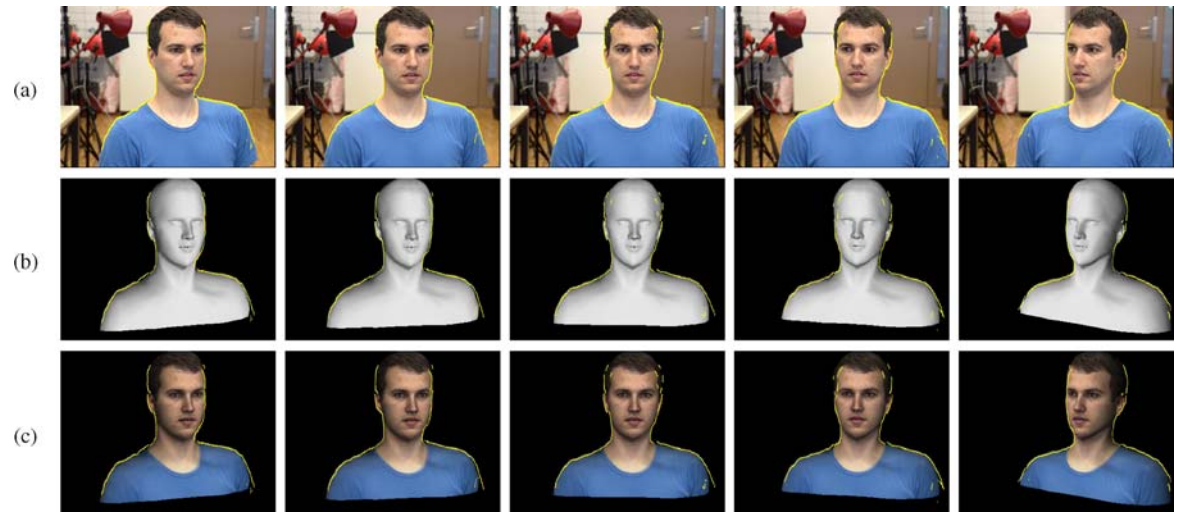


Figure 13. Upper body reconstruction from short uncalibrated video sequence. (a) 5 of 7 images from a short video sequence. (b) Reconstruction result shown as shaded. (c) Texture mapped reconstructed result. The extracted silhouette edges are overlaid on the images.

he wears in the videos of Figs. 14 and 15 in which we track his motions. Again, the occluding contours of the model closely match the true silhouettes. Our technique is robust enough so that, even though the background is very cluttered, we did *not* have to use a background image, that is one without the subject, to make our algorithm's task easier. As shown in bottom row of Fig. 15, the model can be used to resynthesize the subject in front of a different background.

6.2.2. Tracking a Piece of Paper. Our final set of examples involves the tracking of a piece of paper using a coarse rectangular mesh. This time, it has sufficiently few vertexes so that we can parameterize its shape directly in terms of their 3-D coordinates. Let v_i be the coordinates of vertex i, N_v be the set of all pairs of neighboring vertexes, and N_t be the set of triplets of neighboring vertexes aligned along a horizontal or vertical line. To keep the paper's deformations physically plausible, we write the E_D regularization term of

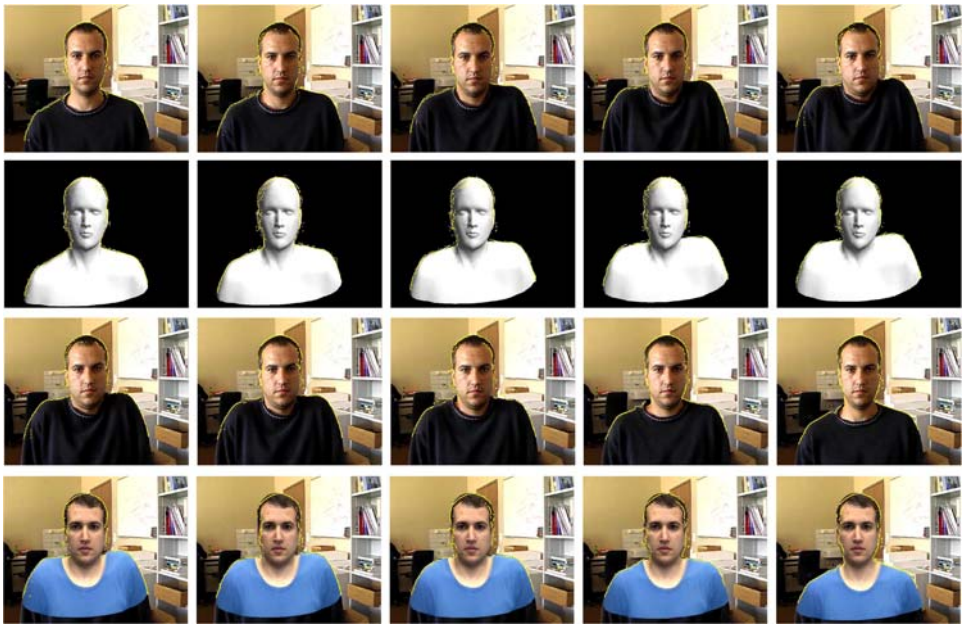


Figure 14. Tracking the subject's rising shoulders in a 61-frame sequence. Top and third row: Images from the original video sequence with extracted silhouette edges overlaid. Second row: Recovered model's shape shown as the shaded model in the same position as the original images above. Bottom row: Recovered model's shape shown as the textured model in the same position as the original images above. Note that silhouette edges are correctly fitted.

Figure 15. Tracking of the moving head and shoulders in a 125-frame sequence. Top row: Images from the original video sequence with detected silhouette edges overlaid. Second row: Recovered model's shape shown as the shaded model in the same position as the original images above. Bottom row: Recovered model placed in front of a different background.

Eq. (12) as

$$
\begin{aligned}
E_D &= w_{bend} E_{bend} + w_{ext} E_{ext}, \\
E_{bend} &= \sum_{(i,j,k)\in N_t} \|2v_j - v_i - v_k\|^2, \\
E_{ext} &= \sum_{(i,j)\in N_v} (\|v_i - v_j\| - L)^2,
\end{aligned}
\tag{24}
$$

where L is the distance at rest between neighboring vertexes and $\{w_{bend}, w_{ext}\}$ are user defined weights. E_{bend} models the bending stiffness of the paper by penalizing excessive curvature. E_{ext} represents the paper's inextensibility by preventing large variations of the distance between neighboring vertexes.

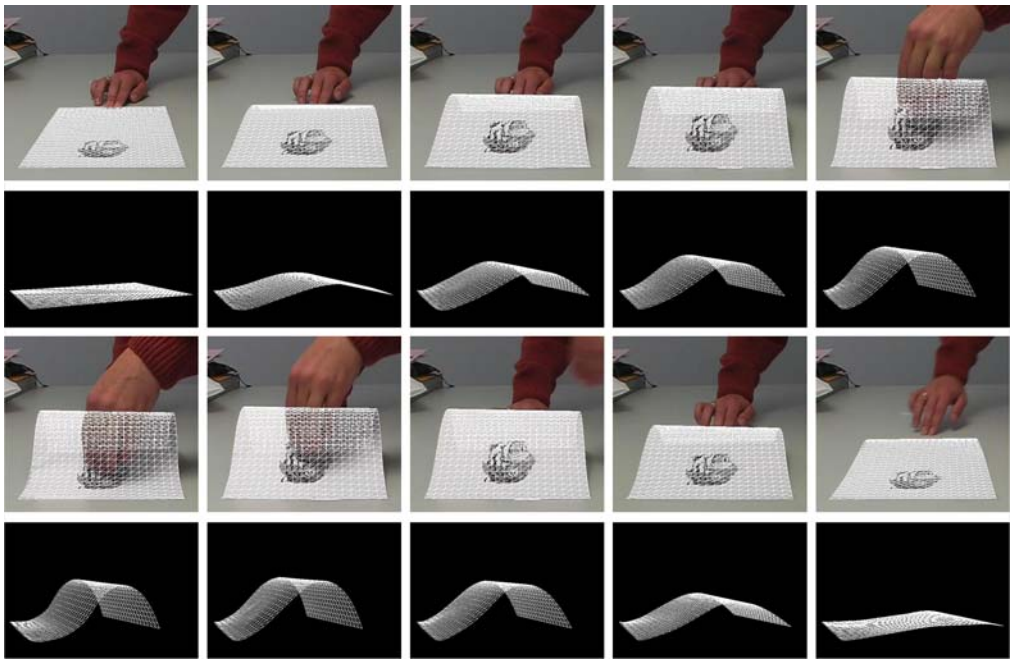


Figure 16. Occlusion handling. The front of the paper is taped to the table and one hand pushes the back of the page while the other passes in front and pushes down. First and third row: The recovered mesh is overlaid on the images. Note that the hand is *in front* of the paper even though the overlay may give the impression that it is behind. Second and fourth row: Corresponding synthetic side-views of the recovered shape. The algorithm is undisturbed by the large occlusion and the flattening of the paper against the table is correctly modeled.

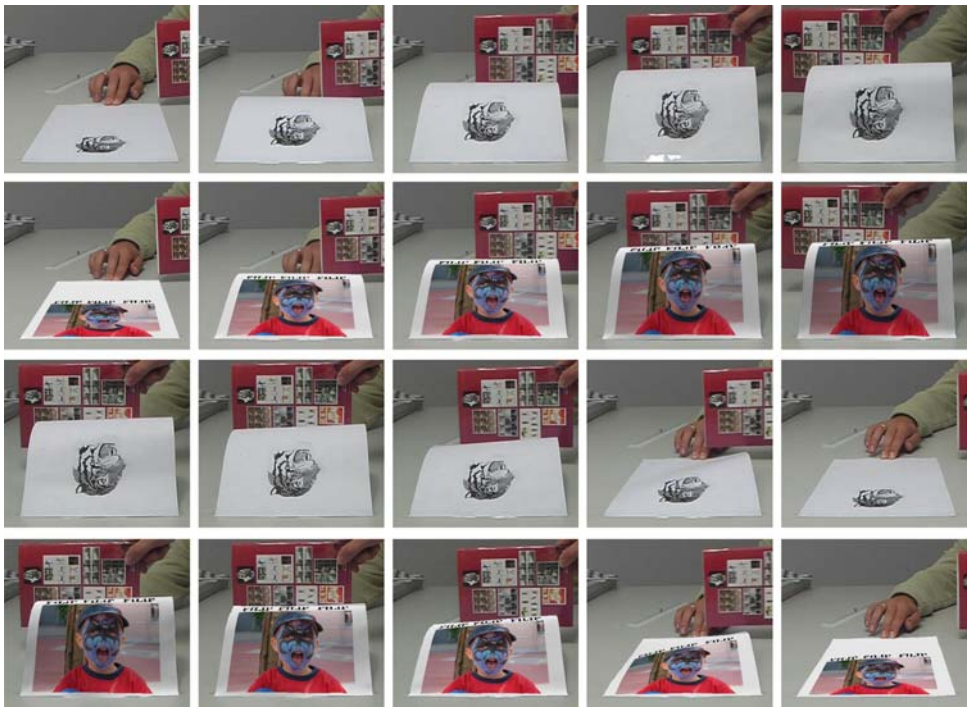


Figure 17. Handling a changing background. Top and third row: Original sequence with a book sliding in the background. Second and bottom row: A new texture is applied on the deformed mesh and reprojected in the images. Note that background subtraction techniques could not have been applied in this case.

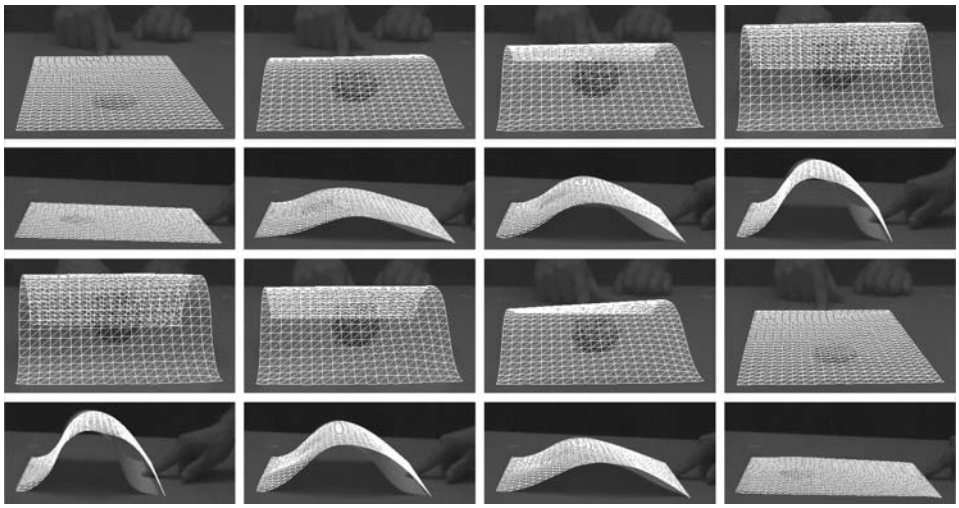


Figure 18. Using side-views to validate our results. First and third row: Front view of the paper used for actual tracking. The recovered model is reprojected as a triangulated wireframe mesh. Second and fourth row: Side images used for verification purposes only. The model is again reprojected onto the images. The front and top of the model accurately matches the paper's outlines. The back of the model is not recovered as well, which makes sense because no shape information is available there.

In the examples of Figs. 16 and 17, the paper initially lies flat on a table and the end closest to the camera is taped to it. It is then pushed from the back. Note that our algorithm automatically detects the apparition of an occluding contour as follows. For each incoming frame, it uses an OpenGL visibility computation to detect whether or not the paper partially occludes itself. In the affirmative, it estimates the 3-D occluding contour using the method of Section 4.2 and matches it to appropriate image edges, as discussed in Section 4.3. The physical borders of the paper are represented by the actual edges of our 3-D model. We also match the projection of the unoccluded ones to nearby image edges and there is no possible confusion between the two kinds of contours. We initialize the process by selecting the four corners of the sheet in the first frame and computing the homography that maps the corners of the initially flat model onto them.

Figure 16 shows that our approach correctly handles the large occlusion created by a hand moving in front of the piece of paper and pushing it down. In the first and third rows, we overlay the recovered model on the original images. In the second and fourth rows, we show corresponding synthetic side-views of the mesh as it deforms. In the left-most image of the fourth row, it can be clearly seen that the front of the sheet of paper is flattened against the table. Figure 17 highlights the robustness of our algorithm to a changing background. The first row shows the original sequence with the same tiger image as before and a moving book behind. In the second row, we used the deformed mesh to map a new texture onto the images. The new images look realistic and, because of the moving book, this result couldn't have been obtained by using a simple background subtraction technique.

Because our laser scanner is not designed to acquire dynamic scenes, we chose to validate these results by filming the deforming piece of paper from the side, using a second synchronized camera. We used a commercially available photogrammetric calibration package to compute the relative orientations of the two cameras, thus allowing us to re-project the reconstructed 3-D models in the side-views. As shown in Fig. 18, even though these side-views were *not* used for reconstruction purposes, the front and top of the model, that is where there is either correspondence or silhouette information, project at the right place while the back does not. This was to be expected since we have no information there and use an oversimplified physical model. If our intention had been to recover the shape of the *whole* sheet of paper, we could of course have also extracted and used the silhouettes from the second sequence, but this was not our purpose here.

7. Conclusion

In this work we have presented a framework for the efficient detection and use of silhouettes for recovering the shape of deformable 3-D objects in monocular sequences. We rely on an implicit surface formalism that lets us look for occluding contours as solutions of an ordinary differential equation and enforce the resulting constraints in a consistent manner.

To demonstrate the range of applicability of our method, we applied it to three very different problems: Reconstructing a PCA based face model from an uncalibrated video sequence; tracking a deforming piece of paper undergoing a partial occlusion or with a changing background; recovering head and shoulders motion in a cluttered scene.

In other words, our implicit surface based approach to using silhouettes is appropriate for uncontrolled environments that may involve occlusions and changing or cluttered backgrounds, which limit the applicability of most other silhouette-based methods. Furthermore, our approach is independent from the way the surface deformations are parametrized, as long as this parameterization remains differentiable.

References

Agarwal, A. and Triggs, B. 2004. 3d human pose from silhouettes by relevance vector regression. In *Conference on Computer Vision and Pattern Recognition*.

Blanz, V. and Vetter, T. 1999. A morphable model for the synthesis of 3-D Faces. In *Computer Graphics, SIGGRAPH* Proceedings, Los Angeles, CA, pp. 187–194.

Boyer, E. and Berger, M.-O. 1997. 3D surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3):219–233.

Canny, J. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6).

Carson, C., Belongie, S., Greenspan, H., and Malik, J. 2002. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038.

Cipolla, R. and Blake, A. 1992. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision*, 9(2):83–112.

Cross, G. and Zisserman, A. 2000. Surface reconstruction from multiple views using apparent contours and surface texture. In A. Leonardis, F. Solina, and R. Bajcsy (eds.), *NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics*, Ljubljana, Slovenia, pp. 25–47.

David, P., DeMenthon, D., Duraiswami, R., and Samet, H. 2002. SoftPOSIT: Simultaneous pose and correspondence determination. In *European Conference on Computer Vision*, Copenhagen, Denmark.

Davis, J.W. and Bobick, A.F. 1998. A robust human-silhouette extraction technique for interactive virtual environments. In *Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, Geneva, Switzerland, pp. 12–25.

Dimitrijević, M., Ilić, S., and Fua, P. 2004. Accurate face models from uncalibrated and Ill-Lit video sequences. In *Conference on Computer Vision and Pattern Recognition*, Washington, DC.

Drummond, T. and Cipolla, R. 2002. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):932–946.

Fua, P. 1999. Using model-driven bundle-adjustment to model heads from raw video sequences. In *International Conference on Computer Vision*, Corfu, Greece, pp. 46–53.

Fua, P. 2000. Regularized bundle-adjustment to model heads from image sequences without calibration data. *International Journal of Computer Vision*, 38(2):153–171.

Gavrila, D.M. and Davis, L.S. 1995. 3-d model-based tracking of human upper body movement: A multi-view approach. In *IEEE International Symposium on Computer Vision*, pp. 253–258. IEEE Computer Society Press.

Hartley, R. and Zisserman, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Ilić, S. and Fua, P. 2002. Using dirichlet free form deformation to fit deformable models to noisy 3-D data. In *European Conference on Computer Vision*, Copenhagen, Denmark.

Ilić, S. and Fua P. 2003. Generic deformable implicit mesh models for automated reconstruction. In *ICCV Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, Nice, France.

Kass, M., Witkin, A., and Terzopoulos, D. 1988. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

Kutulakos, K.N. and Seitz, S.M. 2000. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):197–216.

Moccozet, L. and Magnenat-Thalmann, N. 1997. Dirichlet free-form deformation and their application to hand simulation. In *Computer Animation*.

Paragios, N. and Deriche, R. 1998. A PDE-based level-set approach for detection and tracking of moving objects. In *International Conference on Computer Vision*, pp. 1139–1145.

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. 1992. *Numerical Recipes, the Art of Scientific Computing*. Cambridge, MA: Cambridge U. Press.

Rosten, E. and Tom Drummond. 2003. Rapid rendering of apparent contours of implicit surfaces for realtime tracking. In *British Machine Vision Conference*, vol. 2, pp. 719–728, Norwich, UK.

Rother, C. Kolmogorov, V. and Blake, A. 2004. "GrabCut": Interactive foreground extraction using iterated graph cuts. ACM *Transactions on Graphics*, 23(3):309–314.

Sminchisescu, C. and Triggs, B. 2003. Kinematic jump processes for monocular 3D human tracking. In *Conference on Computer Vision and Pattern Recognition*, vol. I, pp. 69, Madison, WI.

Sullivan, S. and Ponce, J. 1998. Automatic model construction, pose estimation, and object recognition from photographs using triangular splines. In *ICCV '98*: *Proceedings of the Sixth International Conference on Computer Vision*, Washington, DC, USA, pp. 510, IEEE Computer Society.

Sullivan, S. Sandford, L., and Ponce, J. 1994. Using geometric distance fits for 3-d. object modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1183–1196.

Szeliski, R. and Weiss, R. 1998. Robust shape recovery from occluding contours using a linear smoother. *International Journal of Computer Vision*, 28(1):27–44.

Terzopoulos, D. and Metaxas, D. 1991. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:703–714.

Vacchetti, L., Lepetit, V., and Fua, P. 2004. Combining edge and texture information for real-time accurate 3D camera tracking. In *International Symposium on Mixed and Augmented Reality*, Arlington, VA.

Vaillant, R. and Faugeras, O.D. 1992. Using occluding contours for 3D object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.